ดร.ไพศาล สงวนหมู่

ข้อแนะนำในการเขียน

โปรแกรมรับส่งข้อมูลผ่านโมเด็ม

สำหรับท่านที่พอจะคุ้นเคยกับโมเด็มบ้างแล้ว อยากจะรู้รายละเอียดทางด้านเทคนิคเพิ่มเติมเพื่อการพัฒนาโปรแกรมควบคุมโมเด็ม ในกรณีที่ต้องการจะพัฒนาโปรแกรมสำเร็จรูปเพื่อการใช้งาน โปรแกรมสำเร็จรูปที่มีอยู่ในบางครั้งก็ไม่ตรงตามความมุ่งหมายของการใช้งานโดยตรง ก็อาจจะต้องมีการทำโปรแกรมประเภท Tailor Made บ้าง

การจับมือกันระหว่างโมเด็มและซีพียู

ก่อนอื่นเรามาทำความรู้จักการเชื่อมต่อระหว่างซีพียูและตัวโมเด็มเสียก่อน โมเด็มนั้นจำเป็นต้องต่อเข้ากับพอร์ตอนุกรมของเครื่อง สำหรับ PC นั้นพอร์ตอนุครมส่วนมากจะเป็นการสื่อสารแบบ Asynchronouse ตามมาตรฐาน RS232C ซึ่งบางทีก็เรียกว่าพอร์ต RS232 ถ้าหากเป็นโมเด็มประเภทภายในจะรวมกับพอร์ตอนุกรมไว้ให้ด้วย การต่อกับโมเด็มจึงทำได้โดยการผ่านทางพอร์ตอนุกรม

IBM PC นั้นใช้ไอซีเบอร์ 8250 สำหรับทำหน้าที่ควบคุมพอร์ตอนุกรม ถ้าเราศึกษาดูให้ดีจะพบว่า 8250 นั้นมีรีจิสเตอร์อยู่ 2 ตัว สำหรับทำหน้าที่ติดต่อกับโมเด็มโดยเฉพาะคือ MCR และ MSR

MCR ก็คือ Modem Control Register สำหรับทำหน้าที่จับมือกับโมเด็ม โดยมีสัญญาณที่สำคัญคือ

-bit#0 DTR หรือ Data Terminal Ready สัญญาณที่บิตนี้คือสัญญาณที่ปรากฏบนพินที่ 20 ของข้อต่อ RS232C สัญญาณที่จุดนี้เป็นตัวบอกโมเด็มว่า คอมพิวเตอร์ ซึ่งเปรียบเสมือนอุปกรณ์ข้อมูลหรือ Data terminal พร้อมที่จะทำงาน

-bit #1 RTS หรือ Request to send เป็นคำขอส่งข้อมูลจากคอมพิวเตอร์ไปยังโมเด็ม ปกติจะไม่ค่อยได้ใช้กันเท่าไหร่นัก ส่วนมากจะทดสอบแต่เพียงว่าโมเด็มพร้อมทำงานหรือไม่เท่านั้น RTS จะอยู่ที่พินที่ 4 ของข้อต่อ RS232

MSR หรือ Modem Status Register เป็นรีจิสเตอร์สำหรับคอยรายงานให้คอมพิวเตอร์ได้ทราบถึงสถานภาพของตัวเอง โดยมีสถานภาพที่รายงานคือ

-bit #4 CTS หรือ Clear to send เป็นสัญญาณบอกคอมพิวเตอร์ว่าพร้อมที่จะรับข้อมูลจากคอมพิวเตอร์ สัญญาณนี้ก็คือสัญญาณที่ปรากฏบนพินที่ 5 ของข้อต่อ RS232

-bit #5 DSR หรือ Data Set Ready เป็นสัญญาณจากโมเด็มไปบอกคอมพิวเตอร์ว่าโมเด็มพร้อมที่จะทำงานแล้ว สัญญาณนี้คือสัญญาณที่ปรากฏบนพินที่ 6 ของข้อต่อ RS232C

– bit #6 Ring indicator เป็นสัญญาณจากโมเด็มส่งไปบอกคอมพิวเตอร์ว่า ขณะนี้มีเสียงกริ่งดังขึ้น สัญญาณนี้ก็คือสัญญาณที่ปรากฏในพินที่ 22 ของข้อต่อ RS232C

-bit #7 Carrier Detect เป็นสัญญาณจากโมเด็มบอกคอมพิวเตอร์ว่าขณะรับสัญญาณพาหะของโมเด็มคู่สนทนาได้แล้ว สัญญาณนี้คือตัวเดียวกับที่ปรากฏในพินที่ 8 ของข้อต่อ RS232

รีจิสเตอร์ MCR จะอยู่ที่ตำแหน่งที่อยู่ของพอร์ตเข้าออกของซีพียูที่ 3FCH สำหรับ COM0: และ 2FCH สำหรับ COM1: MSR จะอยู่ที่ 3FEH และ 2FEH ของ COM0: และ COM1: ตามลำดับ

สัญญาณที่ปรากฏบน MSR จะมีค่าก็ต่อเมื่อสายต่อระหว่างโมเด็มและพอร์ตอนุกรมของซีพียูมีครบถ้วนตามที่บ่งบอกเอาไว้

รูปที่ 1 แสดงถึงสถานภาพของรีจิสเตอร์สำหรับการควบคุม 6 ตัวของ 8250

คราวนี้เราจะเห็นว่าสามารถ

"รายละเอียดทางด้านเทคนิคเพิ่มเติม เพื่อการพัฒนาโปรแกรมควบคุมโมเด็ม ในกรณีที่ต้องการจะพัฒนาโปรแกรมสำเร็จรูปเพื่อการใช้งาน"

ตรวจสอบสถานภาพของโมเด็มได้ตลอดเวลา โดยการอ่านค่าจากรีจิสเตอร์ MSR ของ 8250

นอกเหนือจากวิธีการที่ตรวจสอบสถานภาพของโมเด็มที่รีจิสเตอร์ของ 8250 แล้ว เราสามารถที่จะคอยสอบได้จากคำบอกเล่าของตัวโมเด็มเอง เนื่องจากโมเด็มสมัยใหม่เป็นโมเด็มที่ชาญฉลาด (smart modem) สามารถจะตอบโต้กับซีพียูในลักษณะคำพูด หรือในลักษณะของรหัสตัวเลขผ่านทางสัญญาณข้อมูล เช่น กรณีมีกริ่งโทรศัพท์ดัง โมเด็มจะส่งคำพูด 'RING' ไปบอกซีพียู ถ้าหากโปรแกรมของท่านต้องการจะตรวจสอบว่ามีกริ่งโทรศัพท์ดังหรือไม่ ก็อาจจะทำได้ 2 ทางคือ

1. อ่านค่าจาก MSR ของ 8250 หรือ
2. คอยคำว่า "RING" จากโมเด็ม

จะเห็นว่าวิธีที่ 2 นั้นยุ่งยากกว่าแน่นอน เพราะต้องคอยตรวจสอบคำว่า 'RING' จากข้อมูลที่ส่งมาจากโมเด็ม โปรแกรมสื่อเองก็จะต้องแบ่งออกเป็น 2 ขั้นตอน ขั้นตอนแรกเป็นขั้นตอนก่อนการต่อสายเข้ากับโมเด็มฝ่ายตรงกันข้าม ซึ่งก็จะต้องคอยแปรความหมายคำบอกเล่าที่โมเด็มส่งมาให้ ซึ่งก็จะมีคำว่า

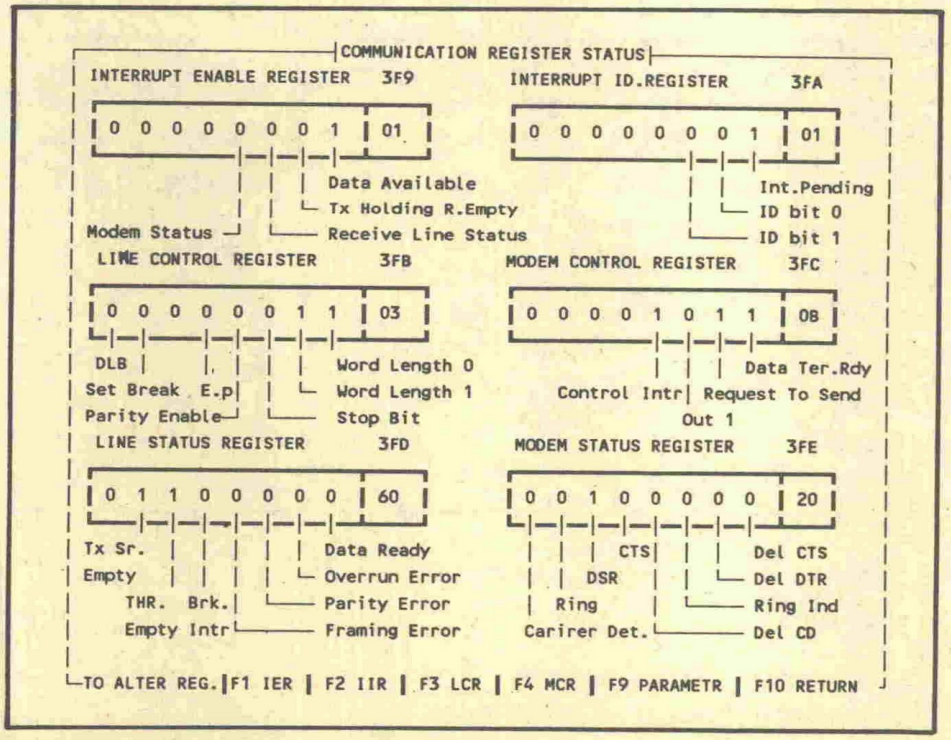
'RING' บอกว่าสัญญาณกริ่งดัง

'ERROR' เมื่อมีการสั่งโมเด็มผิด ๆ

'CONNECT' ต่อโมเด็มฝ่ายตรงกันข้ามได้แล้ว

'NO CARRIER' เมื่อหมุนโทรศัพท์แล้วไม่มีเสียงตอบจากโมเด็มฝ่ายรับ

โปรแกรมจะเข้าสู่ขั้นที่สองซึ่งเป็นขั้นตอนการติดต่อกับคอมพิวเตอร์ฝ่ายตรงกันข้ามเมื่อพบคำว่า 'CONNECT' ในขณะที่อยู่ขั้นตอนนี้ก็จะต้องคอยตรวจสอบคำว่า 'CARRIER LOST' เอาไว้ด้วยในกรณีที่สายการติดต่อเกิดขาดไป จะได้ทำตัวได้ถูกในกรณีเช่นนั้น อันนี้ถือว่าเป็นข้อเสียประการหนึ่งเพราะถ้าหากคอมพิวเตอร์คู่สนทนาเกิดส่งคำว่า 'CARRIER LOST' มา ก็เป็นการตีความผิดแน่ ๆ แต่ถ้าหากตรวจสอบสถานภาพของโมเด็มผ่านทางรีจิสเตอร์ MSR ของ 8250 แล้วจะไม่เกิดปัญหาชนิดนี้เลย



รูปที่ 1 สถานภาพของรีจิสเตอร์ใน 8250

การควบคุมโมเด็ม

พอการจับมือกันระหว่างโมเด็มกับซีพียูเกิดขึ้น การควบคุมโมเด็มก็ทำได้ไม่ยากเนื่องจากโมเด็มรับคำสั่งจากซีพียูผ่านทางข้อมูลที่ส่งมาจากซีพียู ในขณะที่โมเด็มยังไม่ได้เชื่อมต่อกับโมเด็มอีกฝ่ายหนึ่งนั้น ตัวโมเด็มเองก็จะคอยรับคำสั่งจากซีพียูว่าจะให้ทำอะไร คำสั่งที่ส่งไปยังโมเด็มส่วนมากจะนำหน้าด้วย 'AT' ซึ่งย่อมาจาก Attention เป็นการบอกโมเด็มให้สนใจ ลองดูคู่มือการใช้งานของโมเด็มเหล่านั้นจะมีคำสั่งมากมายเกี่ยวกับการควบคุมโมเด็ม แต่ละยี่ห้ออาจจะแตกต่างกันไปบ้างในบางคำสั่งที่พิเศษนอกเหนือจากคนอื่น (ลองดูเรื่องโมเด็มม้าเร็วของคอมพิวเตอร์)

ผมจะขอข้ามขั้นตอนการควบคุมโมเด็มไป เพราะเป็นขบวนการที่ไม่ยากนัก

การแสดงผลของโปรแกรมเทอร์มินัล

ในขณะที่คอมพิวเตอร์เชื่อมต่อเข้ากับศูนย์กลางคอมพิวเตอร์ที่อยู่ระยะไกลในลักษณะที่เป็นเทอร์มินัลนั้น คอมพิวเตอร์ไม่ได้แสดงตัวเองเป็นผู้ทรงปัญญาเลย ไม่ว่าจะเป็นการต่อเข้ากับ Bulletin board หรือ Host computer ก็ตาม คือตัวเทอร์มินัลไม่ได้มีไว้สำหรับคิดหรือประมวลผลใด ๆ นอกเสียจากว่าทำหน้าที่รับข้อมูลหรืออักขระจาก Host computer จับมาแสดงบนจอ และรับข้อมูลจากผู้ใช้ผ่านทางแป้นพิมพ์ส่งไปยัง Host computer เท่านั้น การแสดงผลจึงออกมาในลักษณะของเครื่องโทรพิมพ์คือ บรรทัดต่อบรรทัด ทำอย่างไร Host computer หรือคอมพิวเตอร์ฝ่ายให้บริการจะเข้าไปควบคุมหน้าจอได้ล่ะ เป็นต้นว่าจะเลื่อนเคอร์เซอร์ไปในตำแหน่งที่ต้องการหรือการเขียนแอตทริบิวต์ลงบนหน้าจอเพื่อให้เกิดมีสีสรรที่แตกต่างออกไป หรือว่าการลบหน้าจอให้เหมือนกับโปรแกรมที่ทำงานบนตัวคอมพิวเตอร์เอง ถ้าทำได้ในลักษณะเช่นนี้ก็จะทำให้การแสดงผลบนหน้าจอน่าดูยิ่งขึ้น ในรหัสของแอสกีนั้นไม่ได้มีตัวไหนที่จะสามารถใช้ในการควบคุมหน้าจอได้อย่างมีประสิทธิภาพเลย มีแค่

LF ขึ้นบรรทัดใหม่
CR เลื่อนไปที่ต้นบรรทัด
FF ขึ้นหน้าใหม่
HT แท็ปในแนวราบ
VT แท็ปในแนวดิ่ง

ไม่มีตัวไหนที่จะใช้เลื่อนเคอร์เซอร์ได้เลย

ในกรณีเช่นนี้ได้มีการกำหนดซีเควนซ์ของรหัสพิเศษ เพื่อการควบคุมหน้าจอที่เรียกว่า ANSI.SYS ซีเควนซ์ของรหัสพิเศษเป็นการตกลงทำความเข้าใจกัน ใช้เป็นมาตรฐานในการควบคุมหน้าจอสำหรับเทอร์มินัล IBM PC เองจะเข้าใจโดยตีความหมายของซีเควนซ์ของรหัสพิเศษเหล่านี้ ถ้าหากเราบอกให้เครื่องทำเช่นนั้นโดยการกำหนดไว้ในแฟ้มชื่อ CONFIG.SYS ด้วยประโยคที่ว่า Device = ANSI.SYS และการแสดงผลบนจอของเทอร์มินัลจะต้องเรียกใช้บริการของดอสเท่านั้น เช่นนี้แล้วตัว Host Computer ก็สามารถที่จะเข้ามาควบคุมการแสดงผลของเทอร์มินัลได้

จะขอกล่าวย่อ ๆ ถึงซีเควนซ์ของอักขระพิเศษเหล่านี้ เพื่อว่าจะนำไปเขียนโปรแกรมที่ Host Computer และโปรแกรมที่เทอร์มินัลได้ถูกต้อง

ลักษณะของซีเควนซ์มีรูปแบบดังนี้

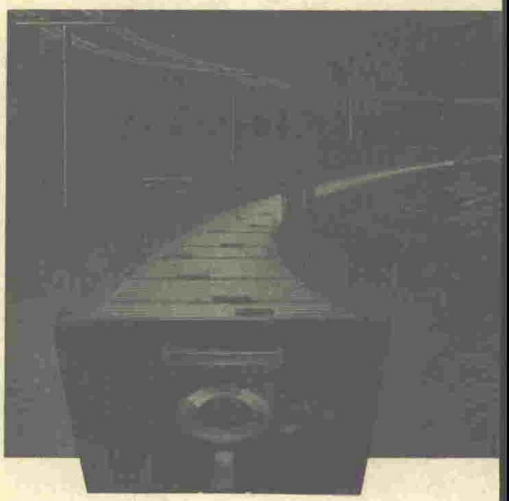
ESC	[	parameter	COMMAND

ESC คืออักขระ 27 ในรหัสแอสกี

[คืออักขระเครื่องหมายวงเล็บเหลี่ยม

parameter คือพารามิเตอร์ที่เป็นตัวเลขใช้ในการคำสั่งที่ตามมา

COMMAND คือสายวลีที่ตามมาบ่งบอกถึงคำสั่ง



รูปที่ 2

ซีเควนซ์	ความหมาย
ESC[#;#H	เลื่อนเคอร์เซอร์ไปยังตำแหน่งที่บ่งบอก # แทนตัวเลข ตัวแรกคือแถว ตัวหลังคือสดมภ์ เช่น ESC[2;10H หมายถึงเลื่อนเคอร์เซอร์ไปที่แถวที่ 2 สดมภ์ที่ 10
ESC[#A	เลื่อนเคอร์เซอร์ขึ้นจำนวน # แถว
ESC[#B	เลื่อนเคอร์เซอร์ลงจำนวน # แถว
ESC[#C	เลื่อนเคอร์เซอร์ไปข้างหน้า # สดมภ์
ESC[#D	เลื่อนเคอร์เซอร์ถอยหลัง # สดมภ์
ESC[2J	ลบหน้าจอ
ESC[K	ลบบรรทัด

ซีเควนซ์สำหรับควบคุมเคอร์เซอร์แสดงรูปที่ 2

ซีเควนซ์สำหรับการควบคุมการแสดงแอตทริบิวต์ (ซึ่งเรียกคำสั่งนี้ว่า SGR หรือ Set Graphic Redition)

ESC[#;#;#m อักขระทุกตัวที่ตามมาจะแสดงแอตทริบิวต์ที่บ่งบอกจนกว่าจะมีคำสั่ง SGR ใหม่ ดังตารางในรูปที่ 3

ซีเควนซ์สำหรับเปลี่ยนโหมดการแสดงผล (SM หรือ Set Mode)

ESC[=#h เปลี่ยนโหมดการแสดงผลใหม่ โดยกำหนดพารามิเตอร์ # ดังรูปที่ 4

หมายเหตุ SM ใช้สำหรับเครื่อง IBM PC เท่านั้น

พารามิเตอร์	ความหมาย
0	เลิกใส่แอตทริบิวต์
1	ตัวเข้ม
4	ขีดเส้นใต้
5	กะพริบ
7	กลับขาว/ดำ
8	ทำให้มองไม่เห็น
30	ตัวหนังสือสีดำ
31	ตัวหนังสือสีแดง
32	ตัวหนังสือสีเขียว
33	ตัวหนังสือสีเหลือง
34	ตัวหนังสือสีน้ำเงิน
35	ตัวหนังสือสีม่วง
36	ตัวหนังสือสีฟ้า
37	ตัวหนังสือสีขาว
40	พื้นสีดำ
41	พื้นสีแดง
42	พื้นสีเขียว
43	พื้นสีเหลือง
44	พื้นสีน้ำเงิน
45	พื้นสีม่วง
46	พื้นสีฟ้า
47	พื้นสีขาว

รูปที่ 3

พารามิเตอร์	ความละเอียด	สีที่ปรากฏ
0	40 × 25	ขาวดำ
1	40 × 25	สี
2	80 × 25	ขาวดำ
3	80 × 25	สี
4	320 × 200	สี
5	320 × 200	ขาวดำ
6	640 × 200	ขาวดำ

รูปที่ 4

โปรโตคอลสำหรับการถ่ายโอนแฟ้ม

แฟ้มข้อมูลถ้ามองในแง่ของการถ่ายโอนข้อมูลก็จะมีอยู่ 2 ชนิด คือ แฟ้มที่เป็นข้อความ (Text File) คือ แฟ้มที่มีเฉพาะรหัสที่เป็นอักขระเท่านั้น ไม่รวมถึงรหัสควบคุมที่นอกเหนือไปจากรหัสควบคุมการแสดงข้อความ เช่น LF, CR, FF, HT, VT ตัวอย่างเช่น แฟ้มที่สร้างขึ้นมาโดย Word Star หรือ Note Pad ของ Side kick แฟ้มอีกประเภทหนึ่งเป็นแฟ้มที่มีข้อมูลที่อาจจะมีค่าได้ตั้งแต่ 0–255 เราเรียกแฟ้มประเภทนี้ว่าแฟ้มเลขฐานสอง (Binary file) ตัวอย่างเช่น แฟ้มคำสั่งที่ลงท้ายด้วย COM หรือลงท้ายด้วย EXE

การรับส่งแฟ้มประเภทข้อความนั้นไม่มีปัญหาในเรื่องที่จะทำให้ตัวโมเด็มเองหรือตัวเทอร์มินัลตีความหมายผิดไป สำหรับแฟ้มประเภท Binary แล้วรหัสข้อมูลภายในแฟ้มมีโอกาสเป็นไปได้ทุกชนิด หากส่งในลักษณะธรรมดาเหมือนกับข้อความอาจจะทำให้เทอร์มินัลตีความหมายผิด ๆ ไปได้ ดังนั้นในการรับส่งแฟ้มข้อมูลไม่ว่าจะเป็นแฟ้มข้อความหรือแฟ้มเลขฐานสอง ทางที่ดีควรจะมีระเบียบกติกาสำหรับการรับส่งที่เป็นที่เข้าใจกันทั้งฝ่ายรับฝ่ายส่ง ระเบียบหรือกติกาที่ว่านี้ก็คือ โปรโตคอลในการรับส่งนั้นเอง

นอกจากจะช่วยในการรับส่งข้อมูลให้เป็นแบบแผนแล้ว โปรโตคอลควรจะต้องมีวิธีการในการ

– ตรวจสอบข้อผิดพลาดจากการรับส่ง

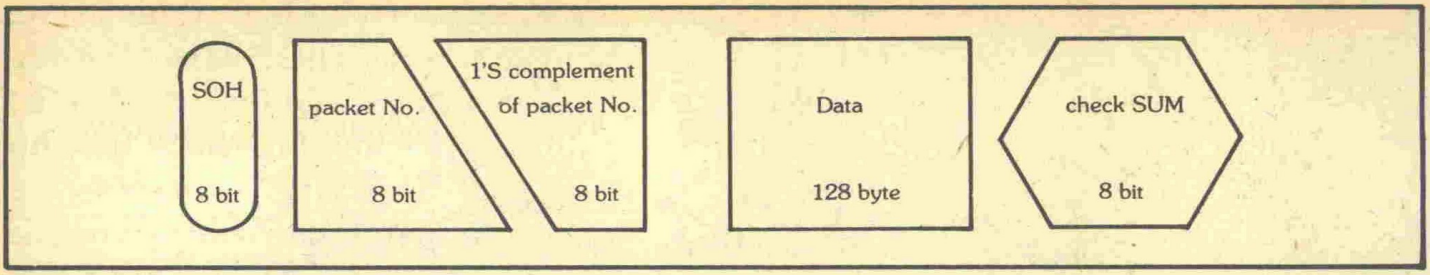
– มีขบวนการที่แน่นอน ที่จะต้องปฏิบัติเมื่อมีข้อผิดพลาดในการรับส่ง

– มีวิธีการเรียงลำดับของกลุ่มข้อมูลที่ส่ง

– มีวิธีการที่ทำให้ข้อมูลประเภทแฟ้มเลขฐานสองไปถึงฝ่ายรับได้

โปรโตคอลในการรับส่งข้อมูลส่วนมากจะมีอยู่ในการสื่อสารอนุกรมแบบ Synchronous เช่น BISYNCH, HDLC, SDLC เป็นต้น แต่เมื่อการรับส่งแบบ Asynchronous มีปัญหาก็จำเป็นต้องมีโปรโตคอลในการรับส่งข้อมูลได้

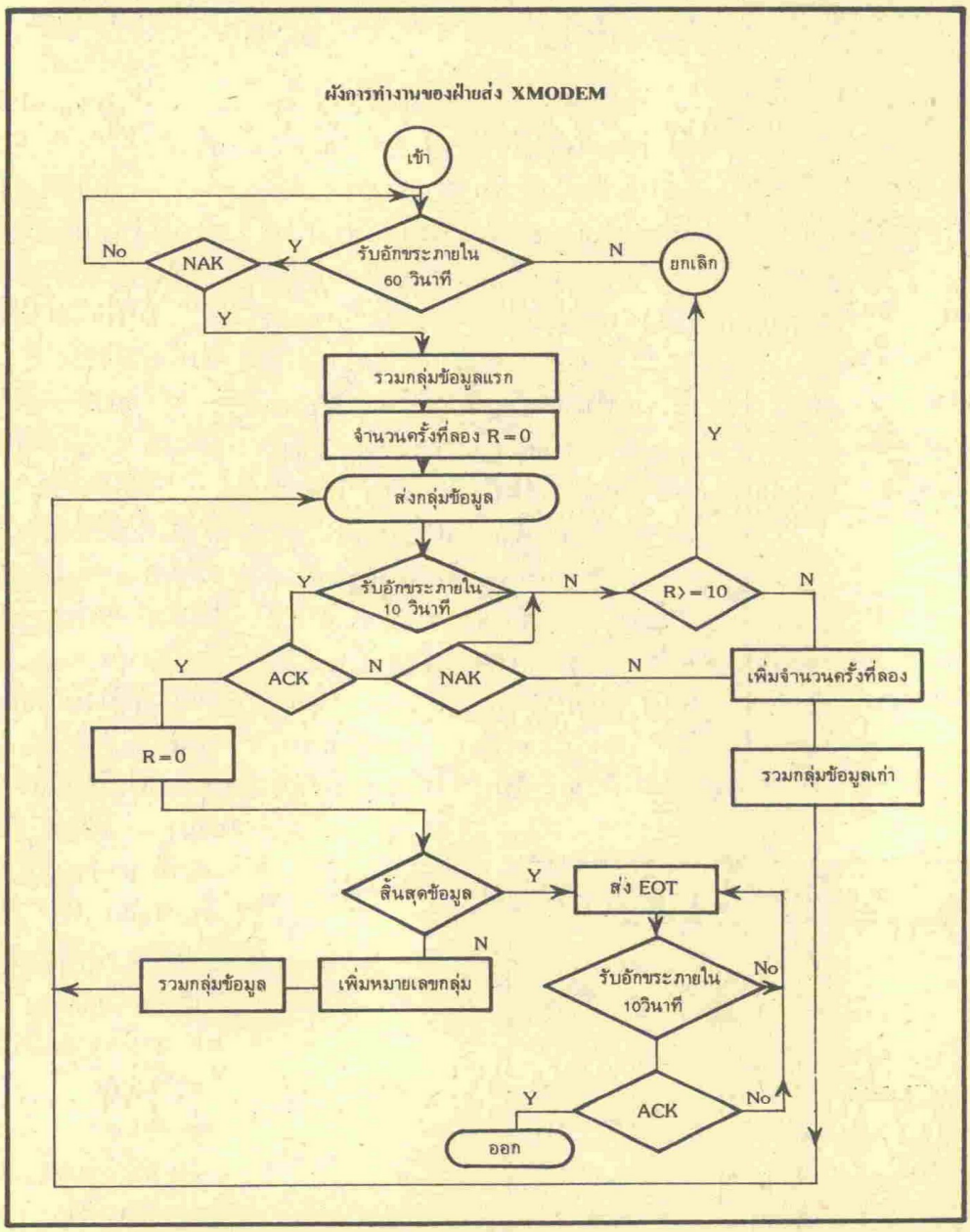
โปรโตคอลสำหรับรับส่งข้อมูลผ่านทางโมเด็มที่รู้จักกันดี คือ XMODEM, KERMIT, YMODEM เป็นต้น XMODEM นั้นมีมาก่อนใครอื่นและเป็นโปรโตคอลตัวแรกที่ใช้ใน Bulletin board และเริ่มใช้ครั้งแรกใน Bulletin board ส่วน KERMIT นั้นพัฒนาขึ้นมาทีหลัง นัยว่าเป็นการเพิ่มประสิทธิภาพในการรับส่งผ่านคู่สายที่มีผ่านระยะทางไกล ๆ เป็นต้น ว่าต้องผ่านข่ายการสื่อสารดาวเทียมได้ดีกว่า XMODEM



รูปที่ 5 โครงสร้างของกลุ่มข้อมูลของ XMODEM

กลุ่มข้อมูล	รหัส	ความหมาย
SOH	รหัส 01	สำหรับบอกจุดเริ่มต้นของกลุ่มข้อมูล
NAK	รหัส 14	สำหรับฝ่ายรับบอกฝ่ายส่งว่าข้อูลตรวจสอบดูแล้วมีการผิดพลาด
ACK	รหัส 06	สำหรับฝ่ายรับบอกฝ่ายส่งว่ารับข้อมูลได้ถูกต้อง
EOT	รหัส 04	ฝ่ายส่งบอกฝ่ายรับว่าหมดข้อมูลที่จะส่งแล้ว

รูปที่ 6



รูปที่ 7 ผังการทำงานของฝ่ายส่ง XMODEM

โปรโตคอล XMODEM

XMODEM รับส่งข้อมูลทีละ 128 ไบต์ ซึ่งเรียกว่า 1 packet โดยมีโครงสร้างของ packet ดังในรูปที่ 5

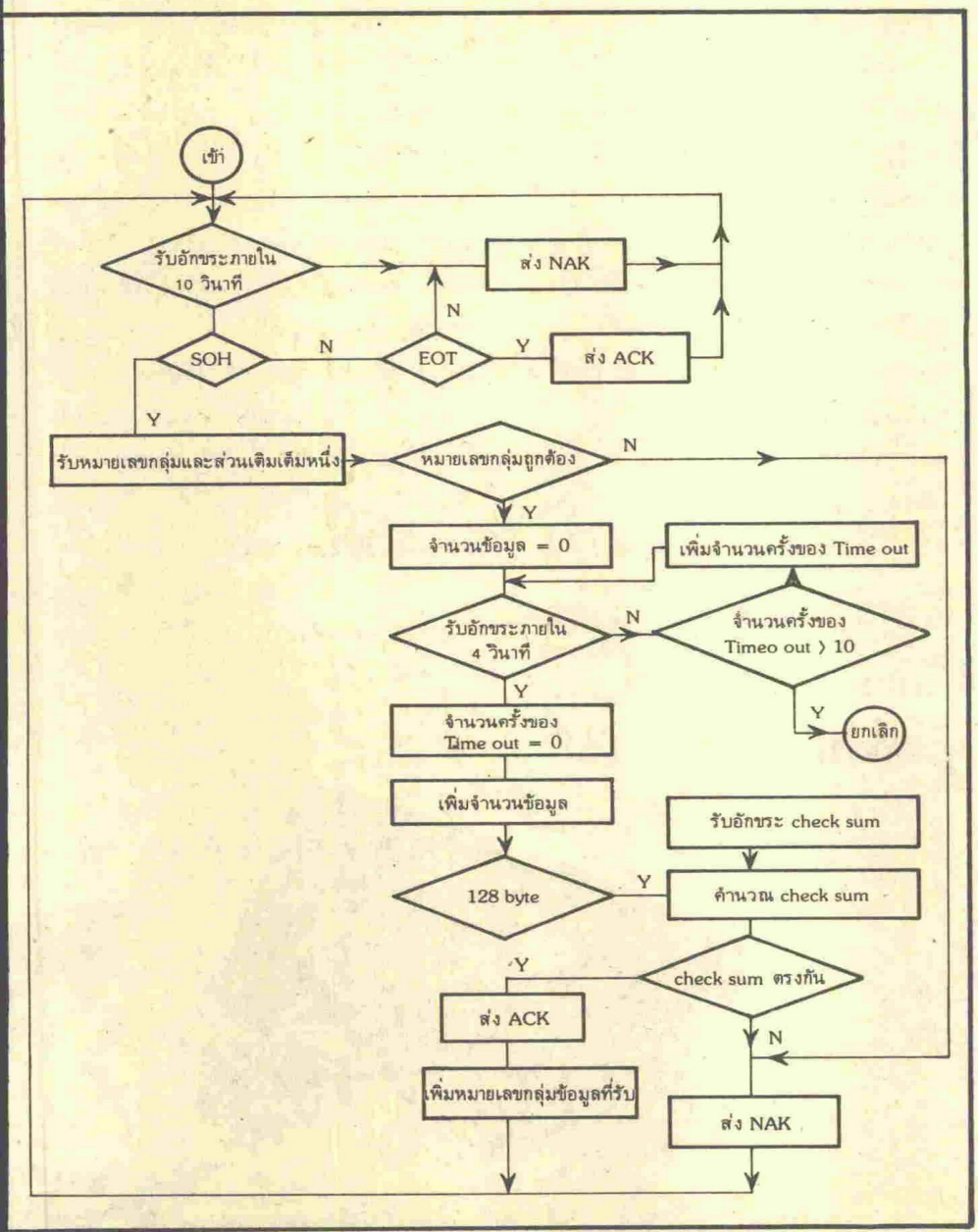
กลุ่มของข้อมูล (packet) เริ่มต้นด้วย SOH (Start of Heading) ซึ่งมีรหัส 1 ในตารางแอสกีตามหลังด้วยหมายเลขกลุ่มข้อมูล 8 บิต เริ่มต้นที่ 1 นะครับไม่ได้เริ่มต้นที่ 0 เพราะฉะนั้นหมายเลขของกลุ่มข้อมูลก็จะเริ่มต้นที่ 1 ไปจนถึง 255, 0 แล้วก็ขึ้น 1 ใหม่ 8 บิตที่ตามมาเป็นส่วนเติมเต็มหนึ่งของหมายเลขกลุ่มข้อมูล เช่น ถ้าหมายเลขกลุ่มเป็น 1 ส่วนเติมเต็มหนึ่งของ 1 ก็คือ 254 หรือ FE ในเลขฐาน 16 การที่มีทั้งหมายเลขของกลุ่มข้อมูลและส่วนเติมเต็มหนึ่งก็เพื่อต้องการตรวจสอบความถูกต้องของหมายเลขกลุ่มข้อมูลเมื่อไปถึงปลายทาง ถ้าทั้งสองไบต์นี้ไม่เป็นส่วนเติมเต็มหนึ่งของกันและกันละก้อ แสดงว่ามีอะไรผิดพลาดแน่นอน

ข้อมูล 128 ไบต์ภายในกลุ่มก็จะตามมา ไบต์สุดท้ายของกลุ่มก็จะเป็นไบต์ตรวจสอบที่เรียกว่า check sum ซึ่งก็คือ ผลรวมของข้อมูล 128 ไบต์ใน Modulo ของ 256 หรือเอาเฉพาะ 8 บิตสุดท้ายของการรวม ถ้าหากฝ่ายรับรับแล้วปรากฏว่ารวมข้อมูลออกมาแล้วตรวจสอบกับ check sum ไม่เท่ากัน ก็แสดงว่ามีข้อผิดพลาดเกิดขึ้นในการรับส่งข้อมูลแล้ว วิธีการตรวจสอบแบบตรวจสอบผลรวมนี้ยังมีจุดอ่อนหลายประการ ท่านคงจะพอนึกออก เป็นต้นว่าเกิดมีการส่งไปแล้วมีสัญญาณรบกวนทำให้ถึงฝ่ายรับข้อมูลผิดไป 2 ไบต์ แล้วก็เผอิญที่ผิดไปรวมกันได้ 256 พอดี ในกรณีนี้การตรวจสอบแบบผลรวมจะใช้ไม่ได้

ในการรับส่งแบบ XMODEM จะใช้รหัสควบคุมในตารางแอสกี ดังรูปที่ 6

จุดอ่อนของ XMODEM ก็คือการที่ฝ่ายส่งต้องรอรับสัญญาณ Acknowledge จากฝ่ายรับทุกครั้งที่ส่งไป 1 กลุ่มข้อมูลทำให้เสียเวลา ถ้าหากระยะทางของการสื่อสารยาวมาก ๆ เป็นต้นว่าข้ามทวีปหรือข้ามประเทศที่ต้องผ่านดาวเทียม delaytime ของสัญญาณมีมาก ประสิทธิภาพของการรับส่งก็จะด้อยลงไป

ดูบล็อกไดอะแกรมทั้งฝ่ายรับและฝ่ายส่งในรูปที่ 7, 8 แล้ว คิดว่าคงจะเขียนออกมาเป็นโปรแกรมไม่ยาก ลองดูซิครับ



รูปที่ 8 ผังการทำงานของฝ่ายรับ XMODEM